



Owning Organizations

**KOREAN
Edition!!!**

SYSCAN 2012

Ryan MacArthur (!backpacker)

SeungJin Lee (beist)



Introduction

- ✧ Beist
 - ✧ bug and beer hunter
- ✧ Mac
 - ✧ wanna-be Korean
 - 당신은 예뻐요

What this talk is not

- ✦ Advanced
- ✦ Persistent

Agenda

- ✦ Hermit Kingdom Software
 - ✦ Even obama knows kakao-talk
- ✦ How it should be done
 - ✦ a case study
- ✦ Efficient bug discovery
 - ✦ native speed fuzzing
- ✦ Real world testing
 - ✦ evading/owning everything

Application Popularity

- ✦ Korea has popular regional applications that have massive adoption rates
 - ✦ most of which are unknown outside of Korea/Koreans

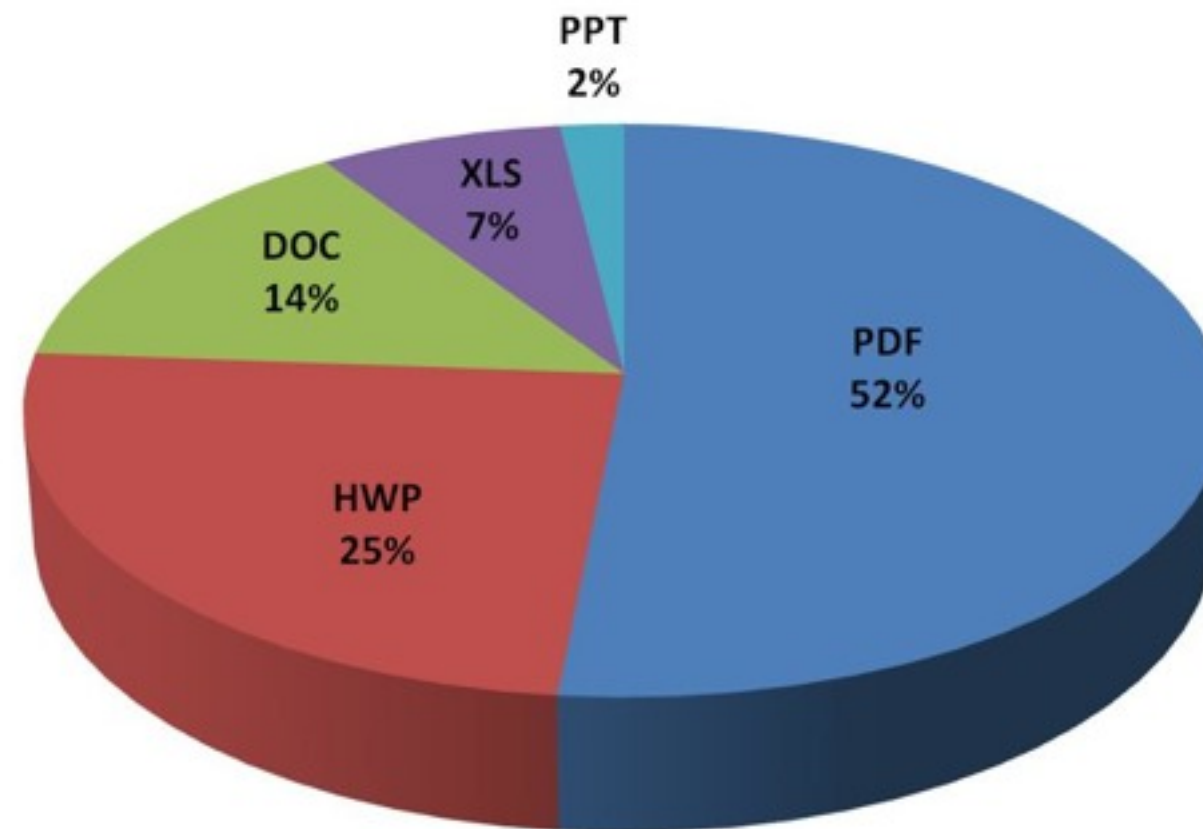
Unique and Popular Software

- ✧ PC
 - ✧ client side
 - ✧ messengers
 - ✧ nateon, daum
 - ✧ security products (AV/Vaccine)
 - ✧ ahnlab v3, hauri Virobot, Alyak
 - ✧ office programs
 - ✧ HWP, Jungum
 - ✧ ActiveX
 - ✧ internet banking, stock trading, e-markets
 - ✧ Utility
 - ✧ altools (alzip,alsee, alsong, alshow, alftp), gomplayer, kmplayer, gomtvtv

Unique and Popular Software

- ✧ Server Side
 - ✧ jeus middleware
 - ✧ Tiberio dbms
 - ✧ ZeroBoard
- ✧ Mobile
 - ✧ Messenger
 - ✧ Kakao-talk, nate-talk, ChatOn, Line
- ✧ Network Device
 - ✧ wibro egg device (backdoored)
 - ✧ home router
 - ✧ IPTIME, ZIO, Anygate, Unicorn

Targeted Attacks, Document File Format Exploits (South Korea, 2011)



※ **HWP** : HWP(Hangul Word Processor) is a Korean word processing application file format, and is very popular in South Korea.

Korean Government

- ✦ Allegedly, 99% of computers have hancorn office installed (HWP)
- ✦ government: ahnlab
- ✦ Military: hauri

HWP

- ✧ ecma script implementation
 - ✧ 'hanscript'
- ✧ U3D
- ✧ Embedded Objects

Lets Own HWP?

- ✦ seems like a good target

olleh WiFi zone 찾기

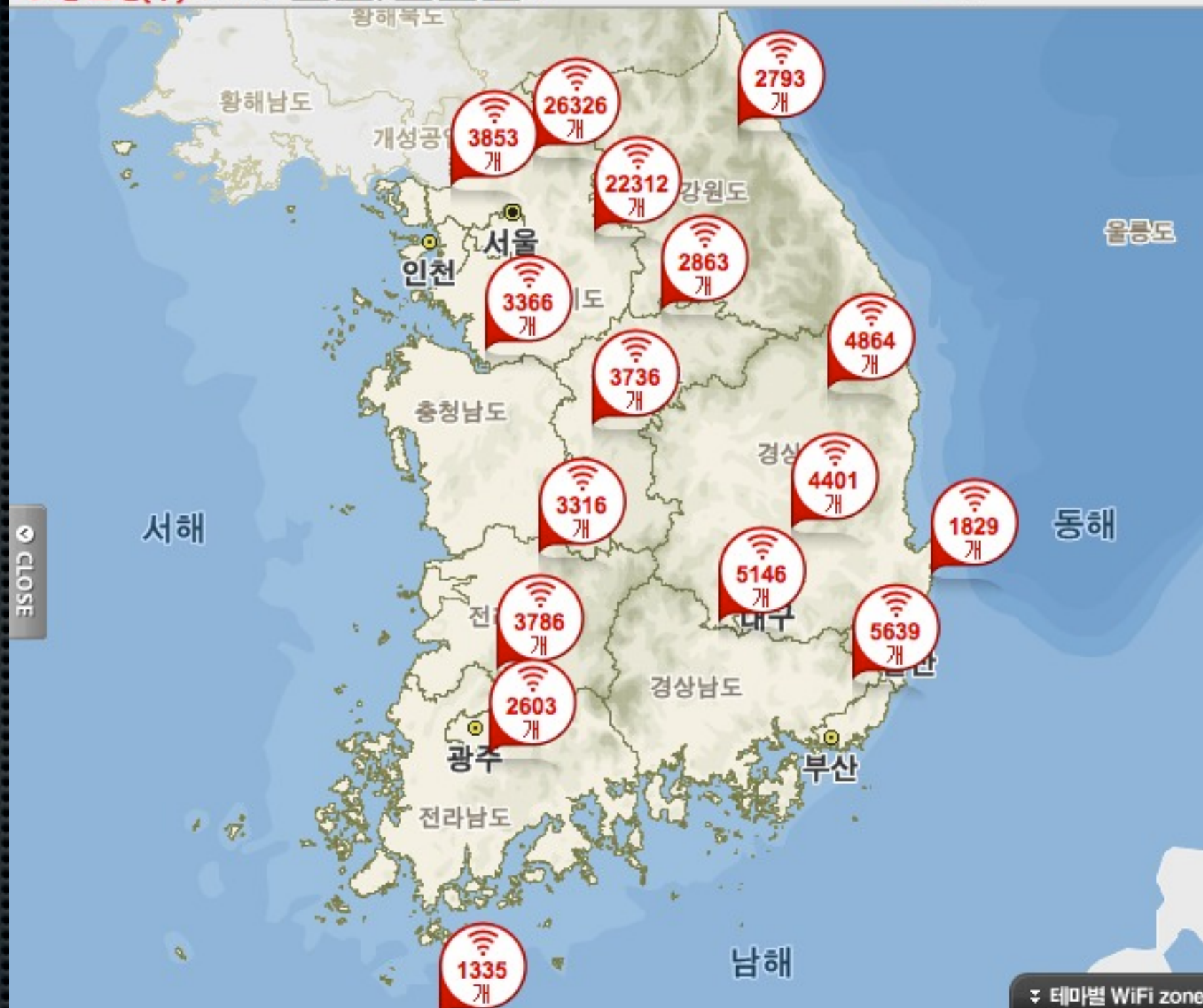
> 오류

전국 olleh WiFi zone

04월 25일(수) 현재 총 98,146 개



가볼 만한 olleh WiFi

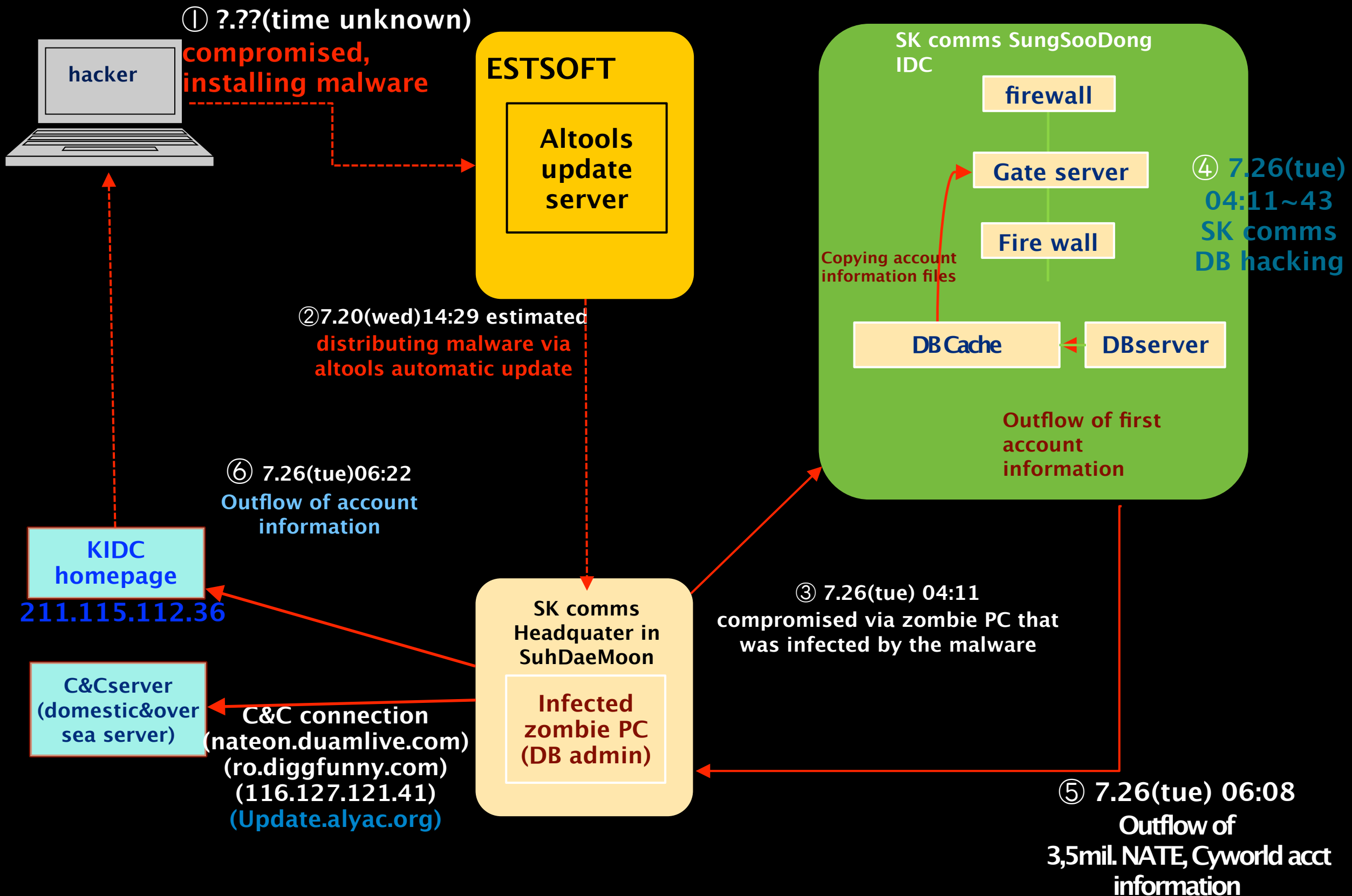


CLOSE

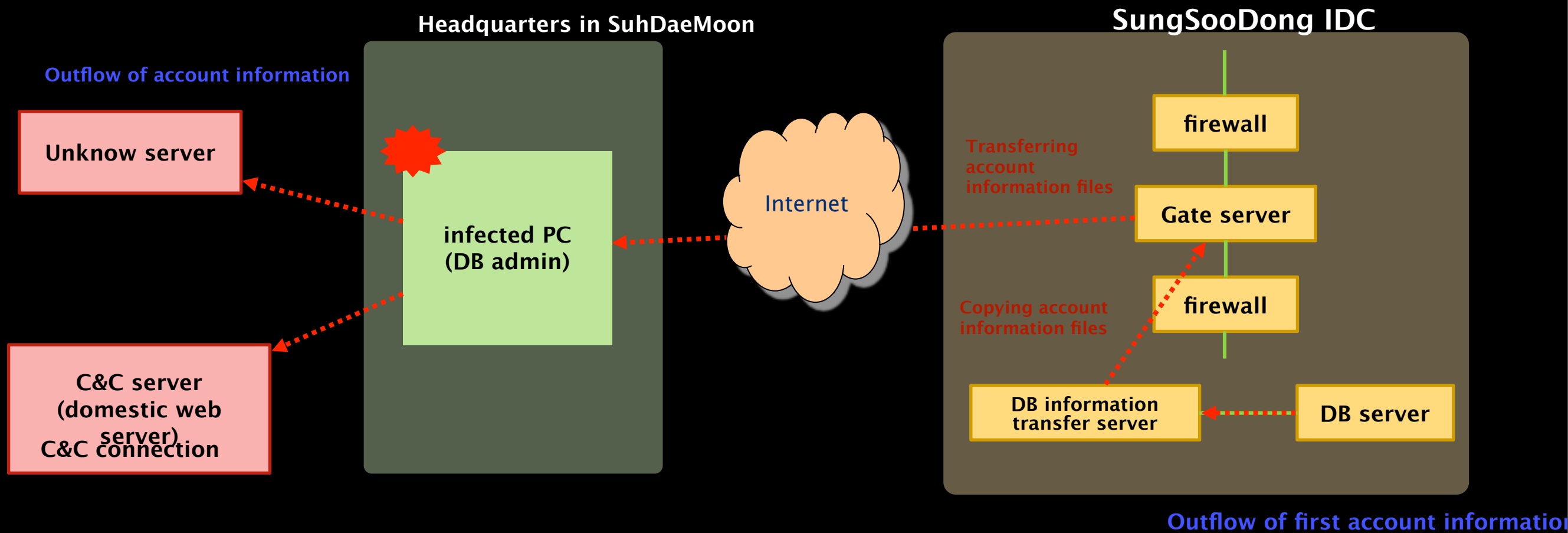
How to steal 35m user records

- July 2011 SK Communications got owned

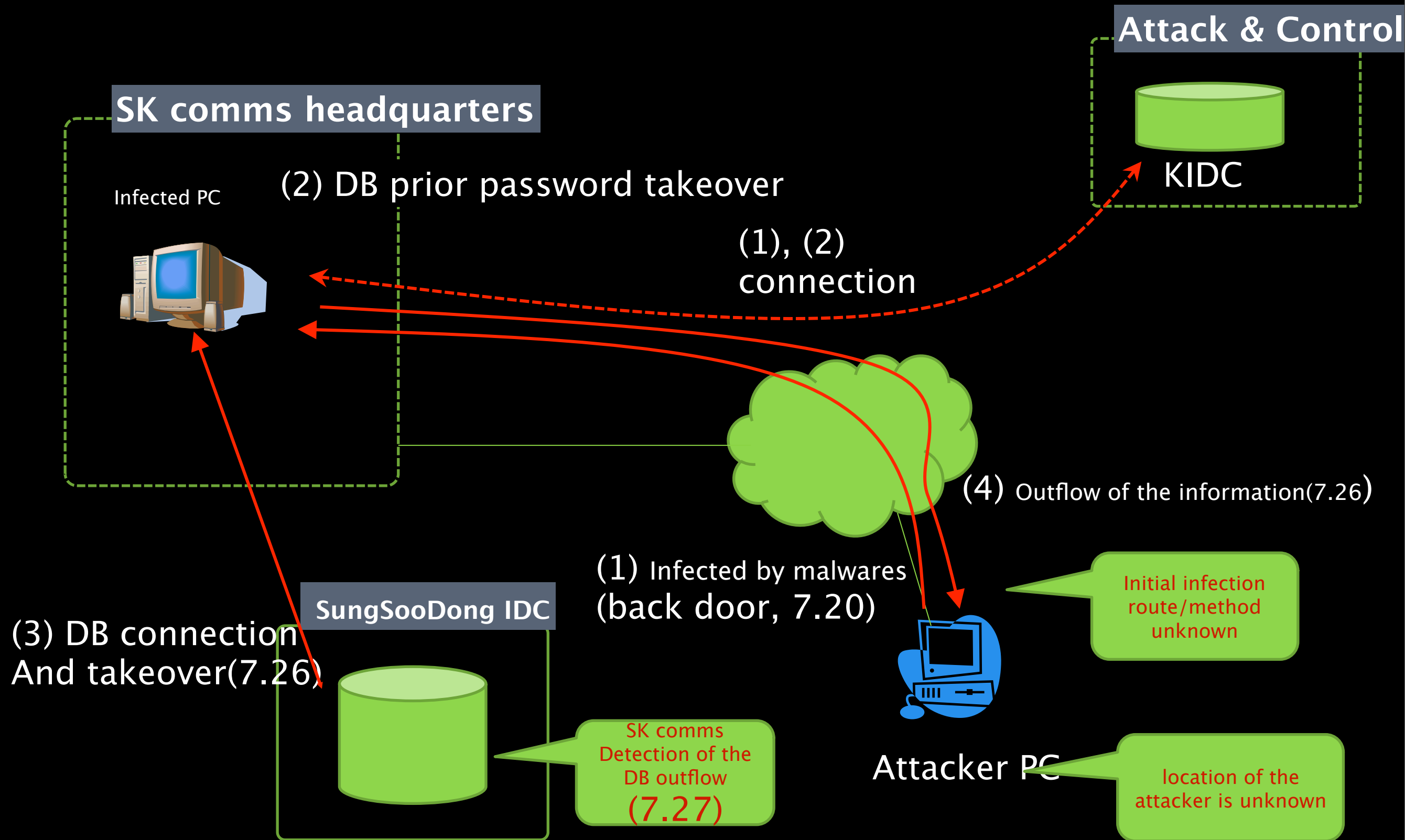
SK Communications Hacking Incident

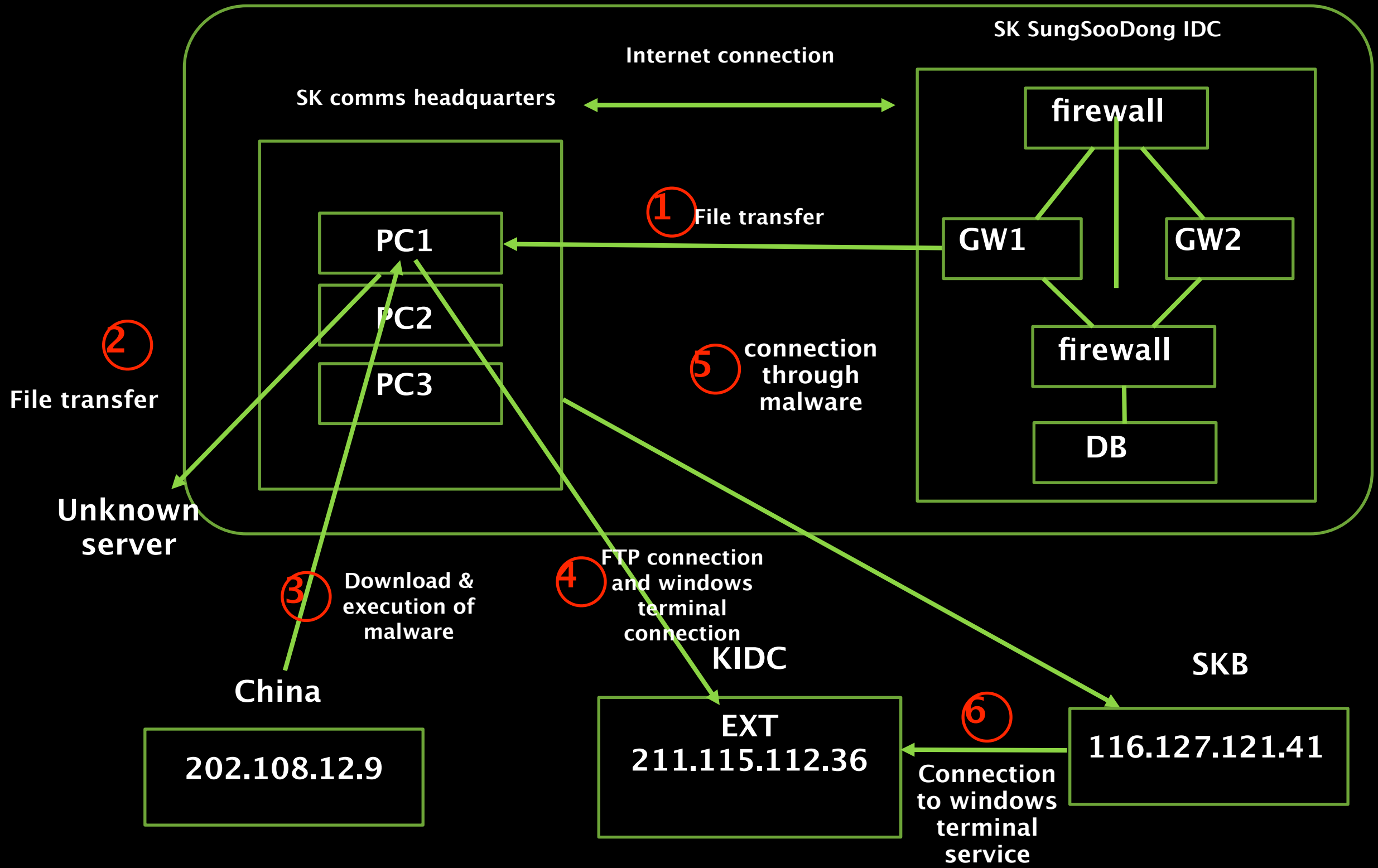


overview of SK comms personal information Outflow



[overview of PII outflow]

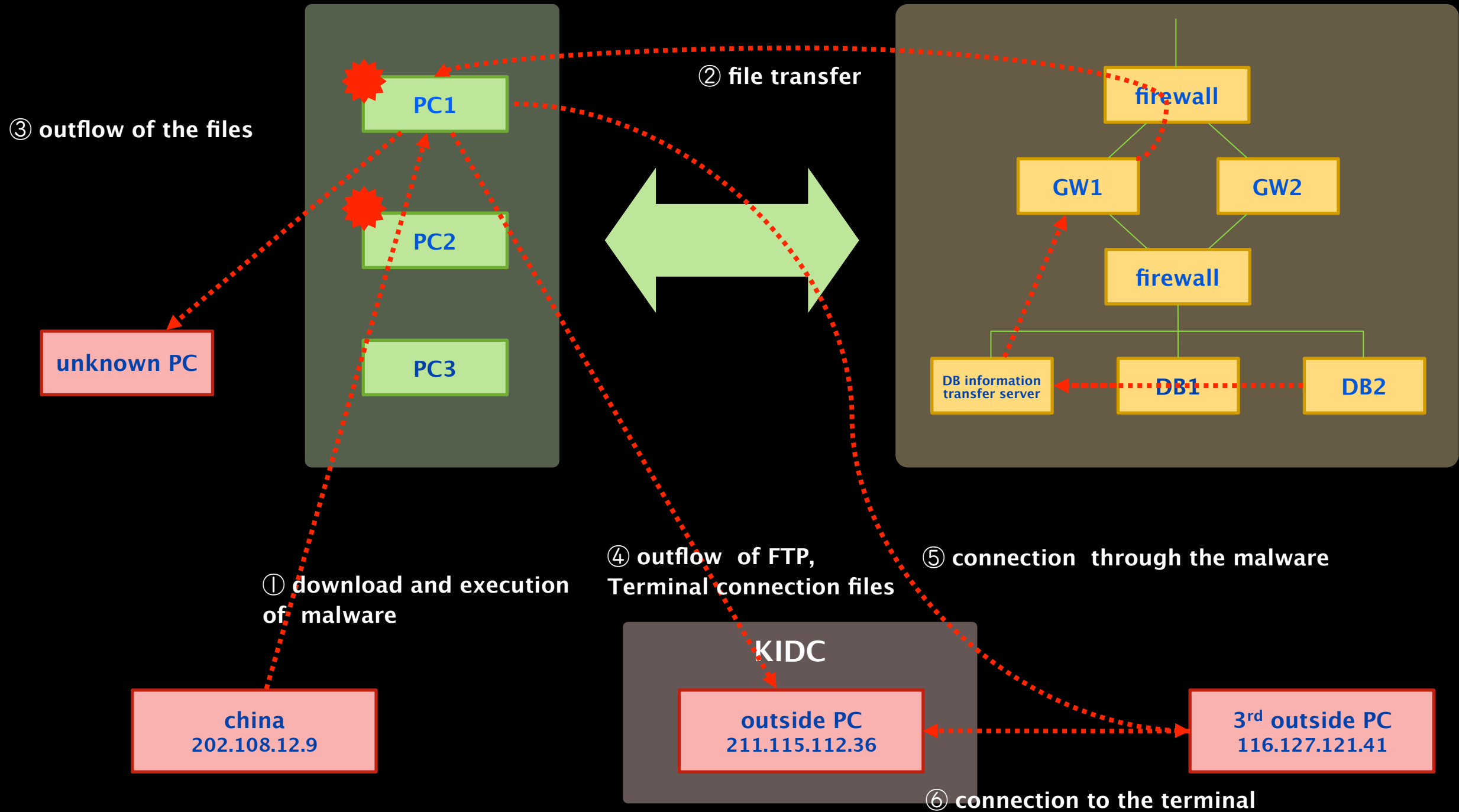




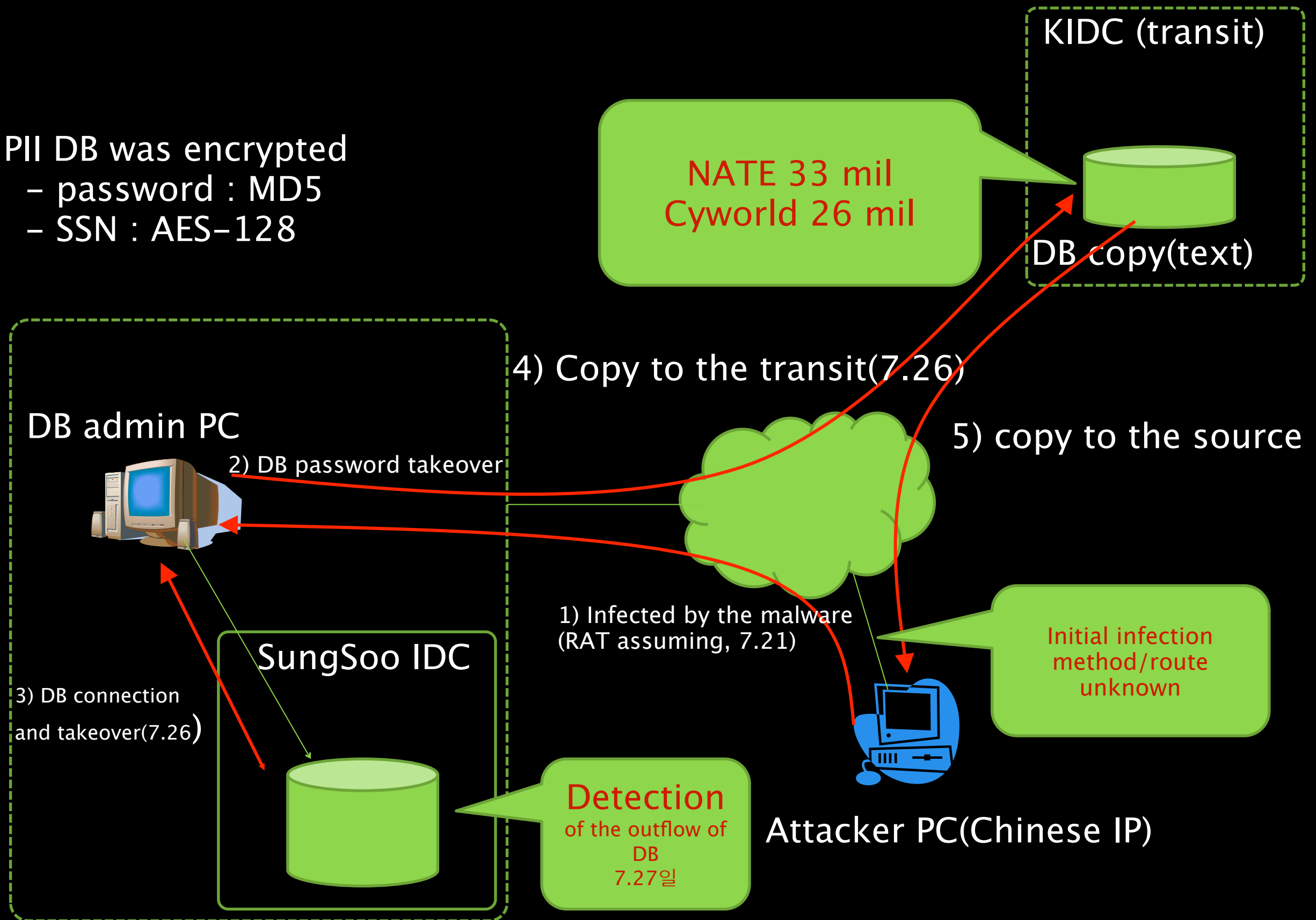
[overview of PII outflow]

SK comms headquarters

SK sungsoodong IDC

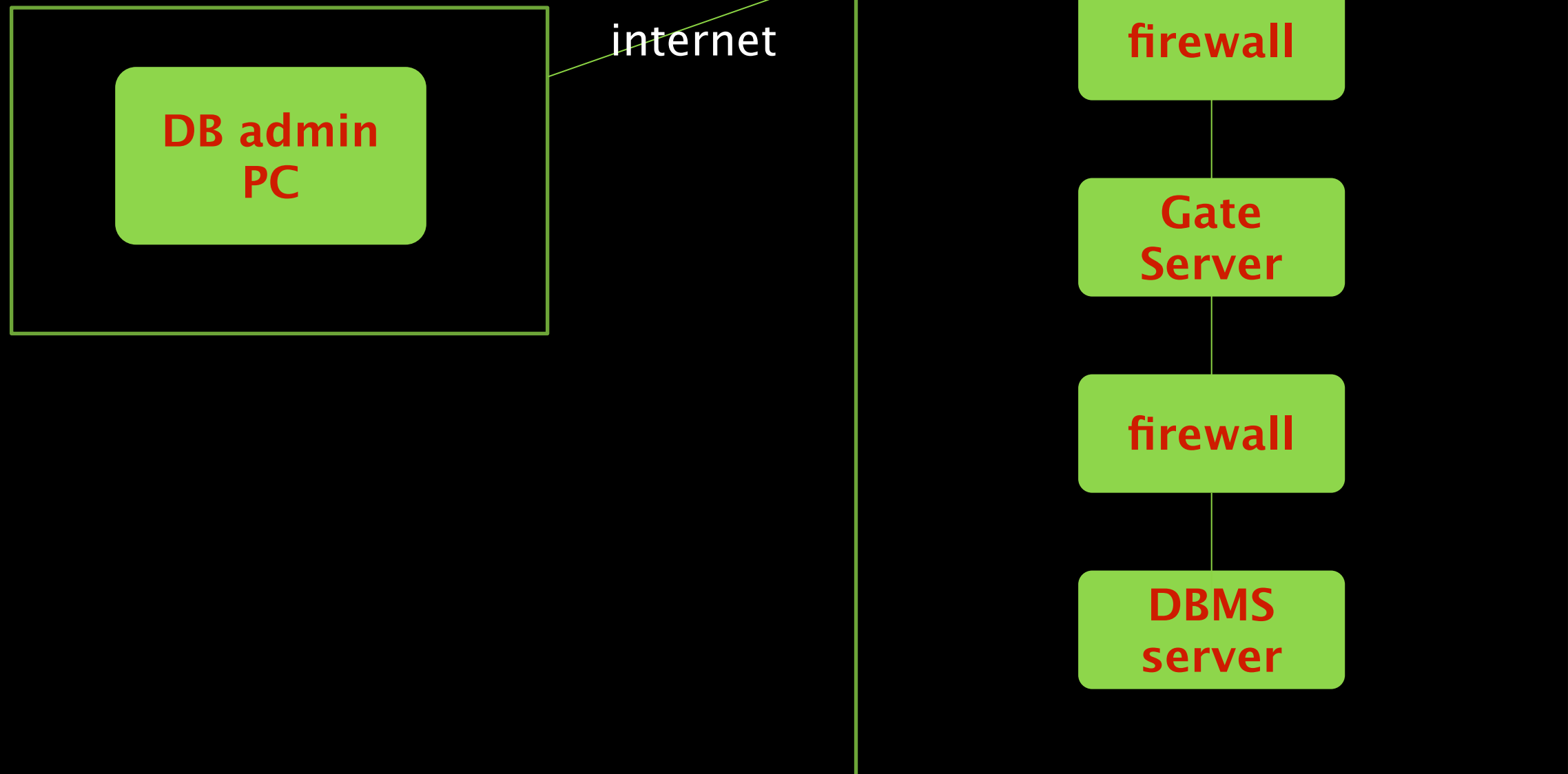


PII DB was encrypted
– password : MD5
– SSN : AES-128



Sungsoo IDC

SK headquarters



Lessons Learned

- ✦ Throttle your exfiltration
 - ✦ try to use WWAN?
- ✦ obfuscate the exfiltration
- ✦ 6 DAYS!

Efficient Bug Discovery

- ✦ Native speed fuzzing + better coverage using recording and comparing

- ✦ Just another talk about file format fuzzing
- ✦ Native speed fuzzing by modifying binaries
- ✦ Better coverage by recording and comparing flows
- ✦ At a immature stage yet, but, a home-brew fuzzer

Motivations

- ✦ There are good tools/theories for fuzzing out there
- ✦ Some are extremely smart but not practical
- ✦ Most of problems are performance issues
- ✦ We wanted to make a practical and not dummy fuzzer

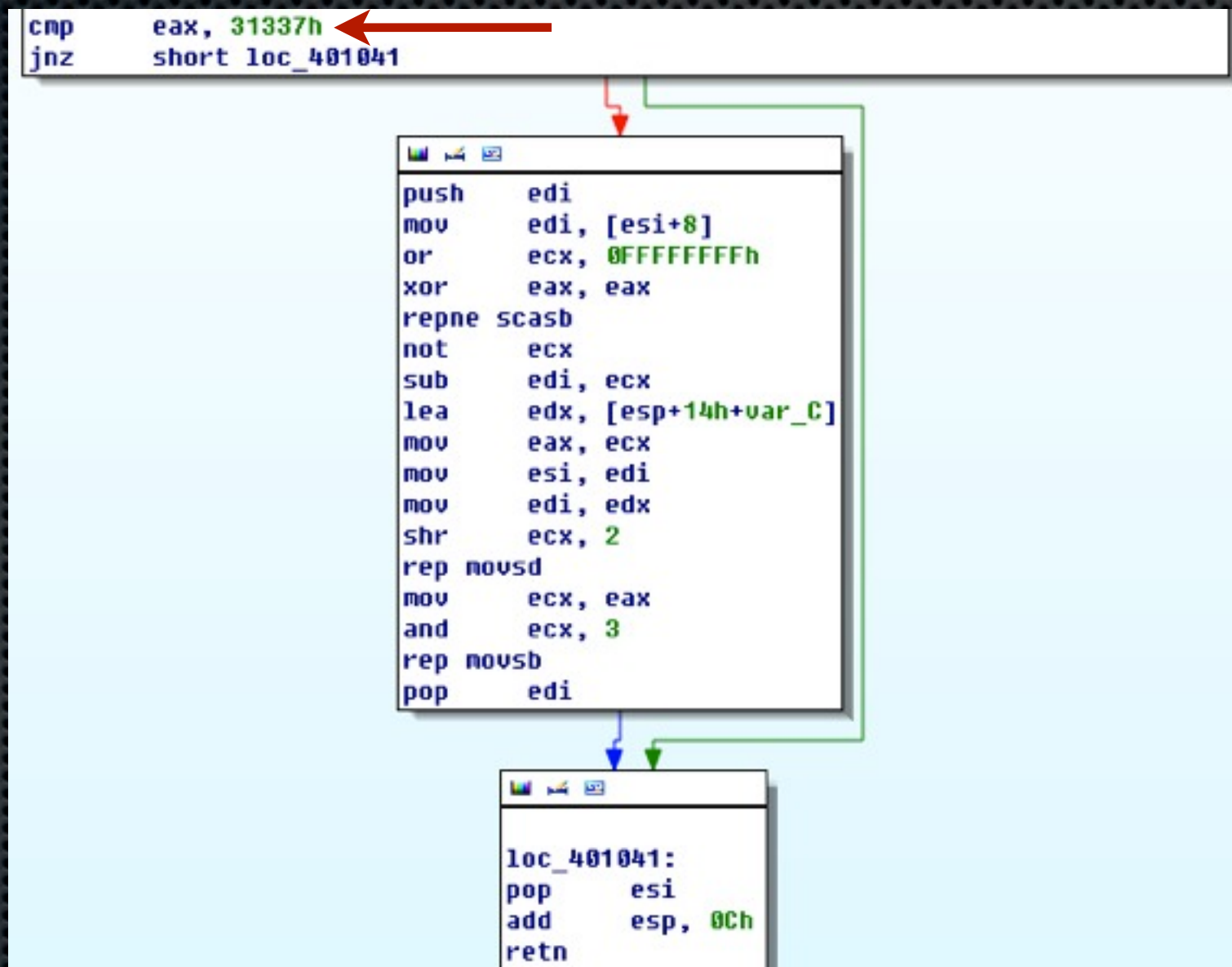
Code coverage

- ✦ More code coverage fuzzing == win (Read grugq's forbes interview!)
- ✦ Every bug hunter who do fuzzing know this problem

```
if (var_a == 0x31337) {  
    strcpy(buf, user_controllable_data);  
}
```

- ✦ How does your fuzzer reach the “**strcpy()**”
- ✦ This is **1/0x100000000** on 32bit

Code coverage



if you set EAX to a value that you want == win

Existing solutions

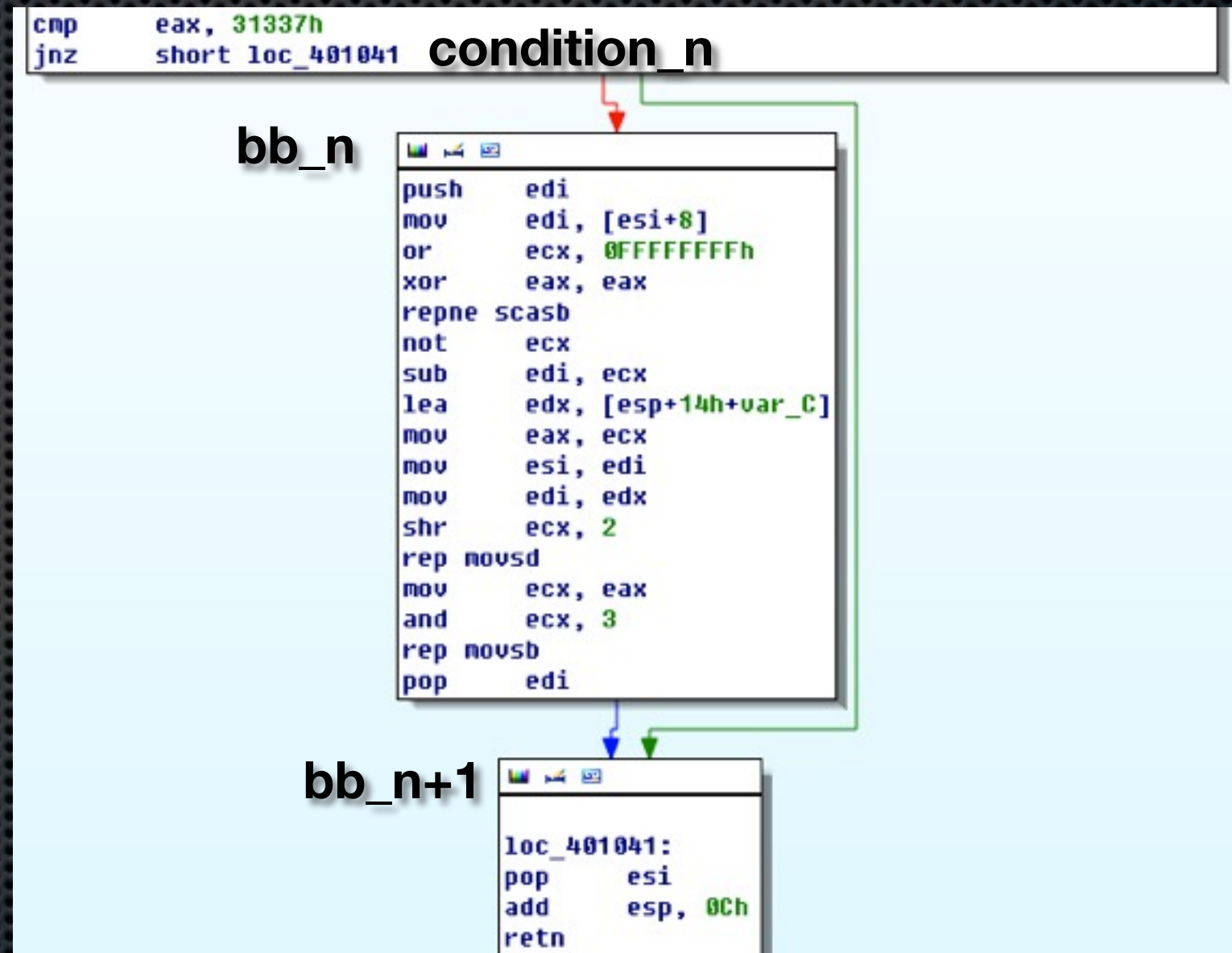
- ✦ There are already clever solutions to solve this problem
- ✦ Intermediate Language, Taint analysis, Symbolic execution, etc
- ✦ But we know if you do fuzzing with these technics, It's slow and costs a lot to develop
- ✦ We can't wait for months until test cases get done

An idea for better coverage

- ✦ The idea is very simple and still developing
- ✦ We record every flow / every test case
- ✦ Also, we collect values of register / memory
- ✦ And compare before - after flows
- ✦ We can evaluate, modify values to get into another flow

Track down flows

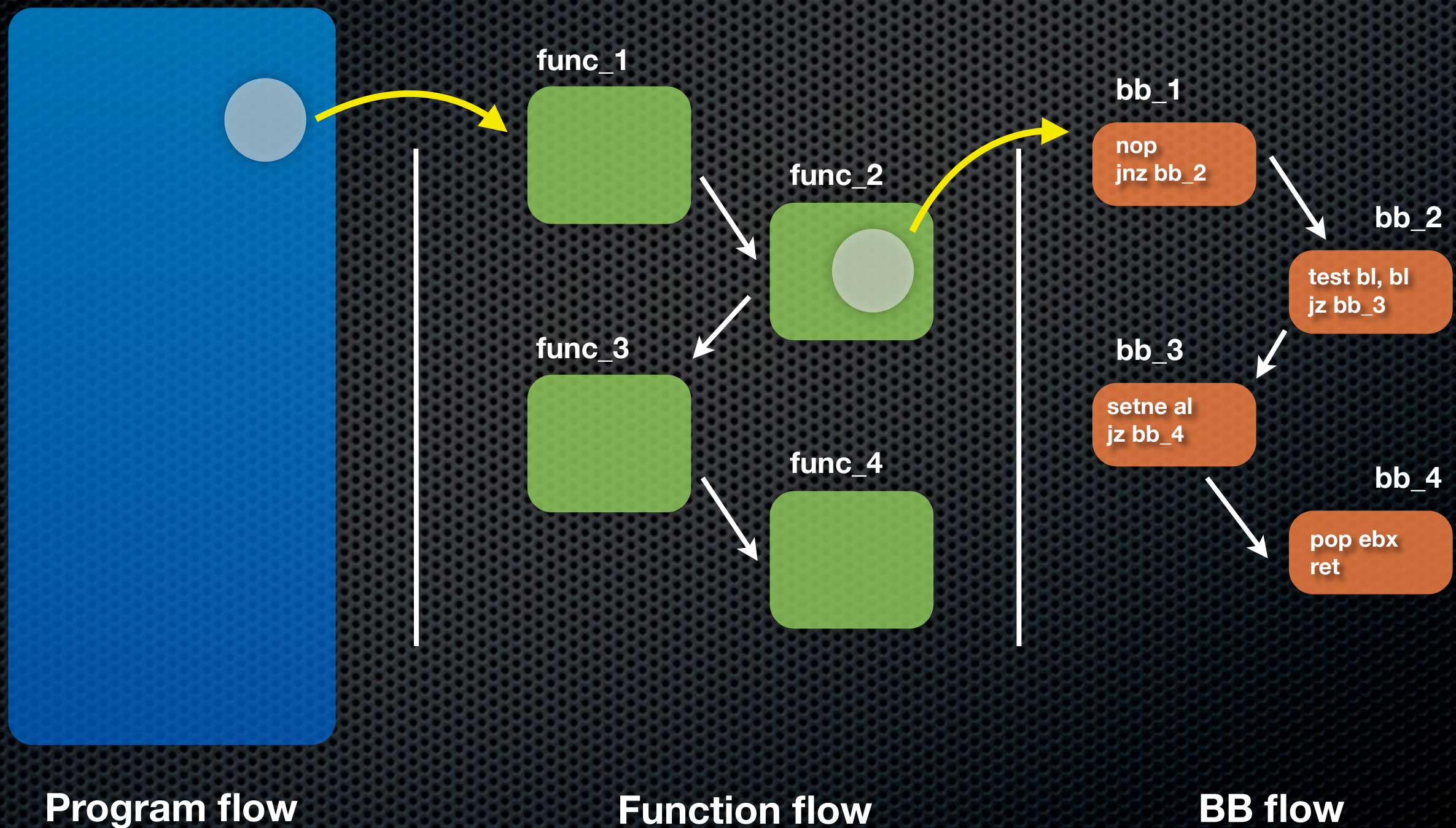
- ✦ Assume the flow gets to *bb_n+1*
- ✦ We find *condition_n*
- ✦ We evaluate how we can get into *bb_n*
- ✦ Figure out we have to change EAX reg (if you change eflags, you'll have many false)



Function level comparison

- ✦ Before we do basic block level monitoring....
- ✦ Your fuzzer never ends if you monitor BB level flows
- ✦ And it would be too many records
- ✦ We should watch function level flows first
- ✦ When we see different flows between 2 cases, we do basic block level monitoring

Deep into..



Function level comparison

- ✦ Let's say we have record_1 and record_2 that have function level flows
- ✦ 2 files are used for fuzzing, just a bit changed
 - ✦ record_1: func_A() - func_B() - func_D() - func_E()
 - ✦ record_2: func_A() - func_B() - func_C() - func_D()
- ✦ We see that the changed data affected flows

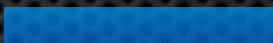
Function level comparison

- ✦ Advantage 1: There are many dummy data in file formats. If we know what data doesn't affect flows, we'll not spend a huge time to fuzz that data area.

[A file]



 - data that don't need to fuzz

 - data that need to fuzz

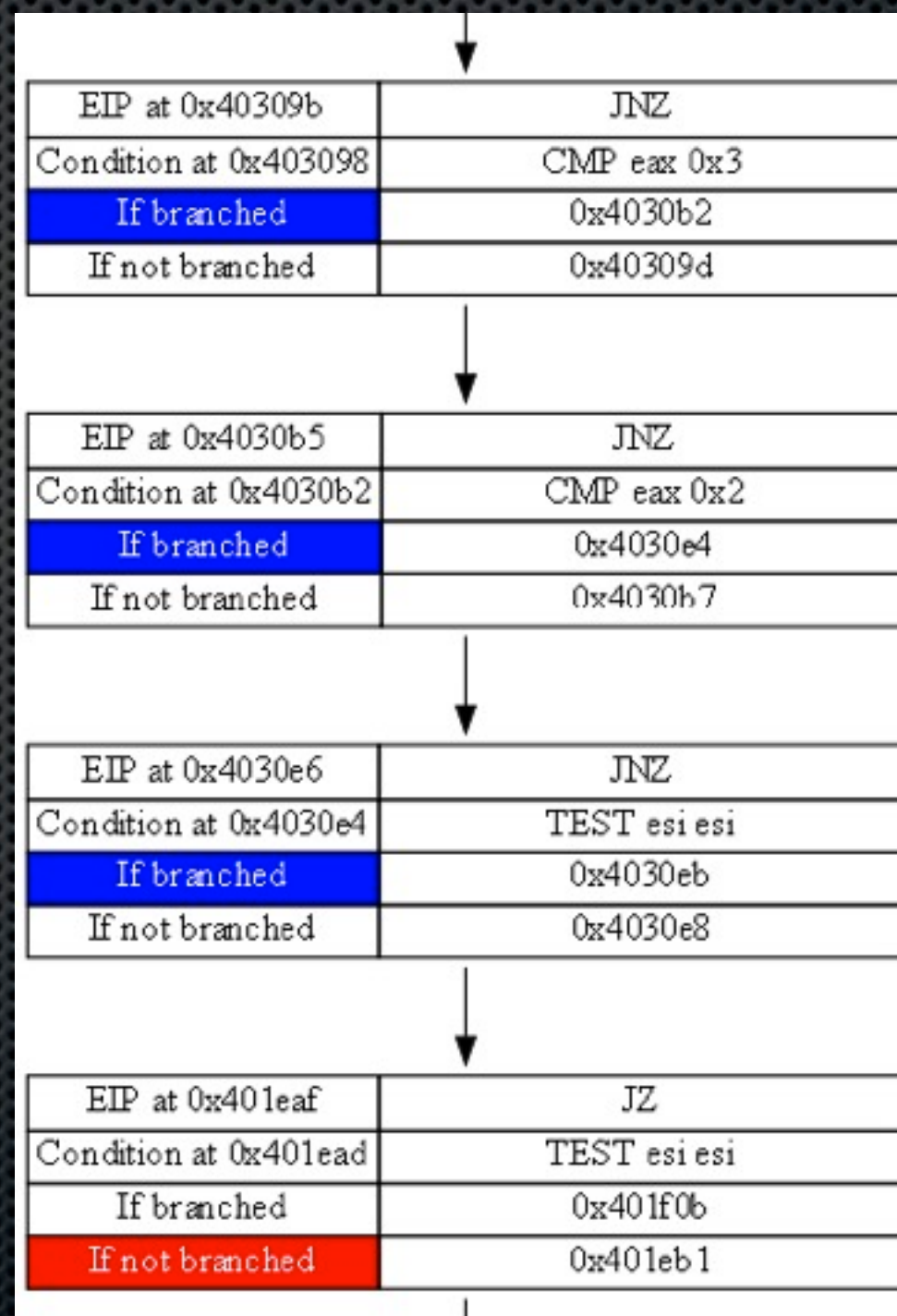
- ✦ Advantage 2: When we see a different function flow, we now go for basic block monitoring. Then, we can evaluate BB flows so that what value is needed to get into another basic block.

Basic block level comparison

- ✦ This is not just recording, not simple
- ✦ You should first know BBs in a function
 - thank you, ida
- ✦ What makes flow different (ex: conditional jumps)
- ✦ You mark every BB_n and figure out why the flow went to BB_x but not BB_y
- ✦ Then, modify your file and fuzz it again to get into BB_y

Basic block level comparison

- ✧ You should be able to match your file and BB's status
 - ✧ **CMP eax, 0x2**
- ✧ Which data in your file is matched to the eax register?
- ✧ After figuring out, change the data of your file and do fuzzing again



Basic block level comparison

- ✦ This idea works very well in simple programs, but..
- ✦ If there are some complex routines like multimedia, compress, crypto, it's very hard to match data between flows and your file
- ✦ We're trying to solve this problem using Symbolic execution (**yet!**)
- ✦ Symbolic execution is slow but doing for specific instructions, so, should be acceptable

Native speed fuzzing

- ✦ ***“Ok.. but i guess it’d be still very slow..? No..?”***
- ✦ Right, if you debug and breakpoint at every function for flow logging, you’d have performance issues
- ✦ It is basically never stopped (hundreds of thousands BPs)
- ✦ So, we threw away debugging and breakpoint stuff
- ✦ We modify binaries itself in order to do flow logging fast

Native speed fuzzing

- ✧ Which means, it runs like native speed (almost)
- ✧ This includes 3 components
- ✧ idapython: extract interesting functions
- ✧ python script: modify functions
- ✧ dll: map a memory area and link it to a file, also adding new features can be easy by dll

IDAPYTHON

- ✧ A script that extracts interesting functions
- ✧ It's easier to say “what is bad” than “what is good”
- ✧ What is a bad function? A function...
- ✧ Has no argument, no loop (or ecx|esi is constant?), no interesting API (strcpy, ++), integer operation mismatches (ja/jb vs jg/jl), any branch, etc
 - ✧ **Should be researched more with security's view**

Python script

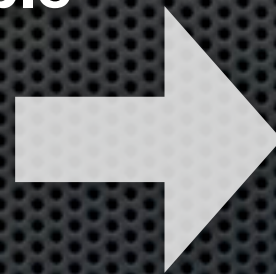
- ✦ 1: Find a garbage(not used) area in a target binary
- ✦ 2: Set a hook at original entry to the garbage area (about 200 byte area is enough)



How garbage area is used

- ✦ 3: Install some code at the garbage area

- 1) routine to find APIs from export table
- 2) VirtualAlloc for 2nd payload
- 3) CreateFile(payload_file)
- 4) ReadFile(payload_data)
- 5) Copy_payload_data_into_memory



garbage area

This works like a virus

Make hashes and set hooks

- ✦ 4: Make a hash file and set hooks at functions
- ✦ 4-1: Read extracted functions which is by idapython
- ✦ 4-2: Get first 12 bytes of functions
- ✦ 4-3: Generate a MD5 hash for 12 bytes
- ✦ 4-4: If the MD5 is new, insert it into a dictionary variable and write the 12 bytes into a file (this is a hash map)

Make hashes and set hooks

- ✦ 4-5: Get a hash number, set a hook at a function like

```
push my_hash_number  
mov eax, 2nd_payload_address  
call eax
```

- ✦ 4-6: This hash map file is loaded by a dll at runtime

Generate 2nd payload

- ✦ 5: Generate 2nd payload file
- ✦ 5-1: VirtualProtect() to modify text code area to restore
- ✦ 5-2: LoadLibrary(our_dll) to load the hash map file
- ✦ 5-3: Restore the entry point and do pro/epilogue stuff
- ✦ 5-4: Restore function's 12 bytes by referencing the hash map file which is mapped on memory

Generate 2nd payload

- ✦ 5-5: Logging this function address
- ✦ 5-6: Back to the function after restoring
- ✦ **dll**: a dll is loaded at runtime. it loads a hash map file, also, we can add any feature easily through this dll

Conclusion

✦ **Advantages**

- ✦ Much better than dummy fuzzing for code coverage
- ✦ Native speed, better than dummy fuzzing (no debug)

✦ **Disadvantages**

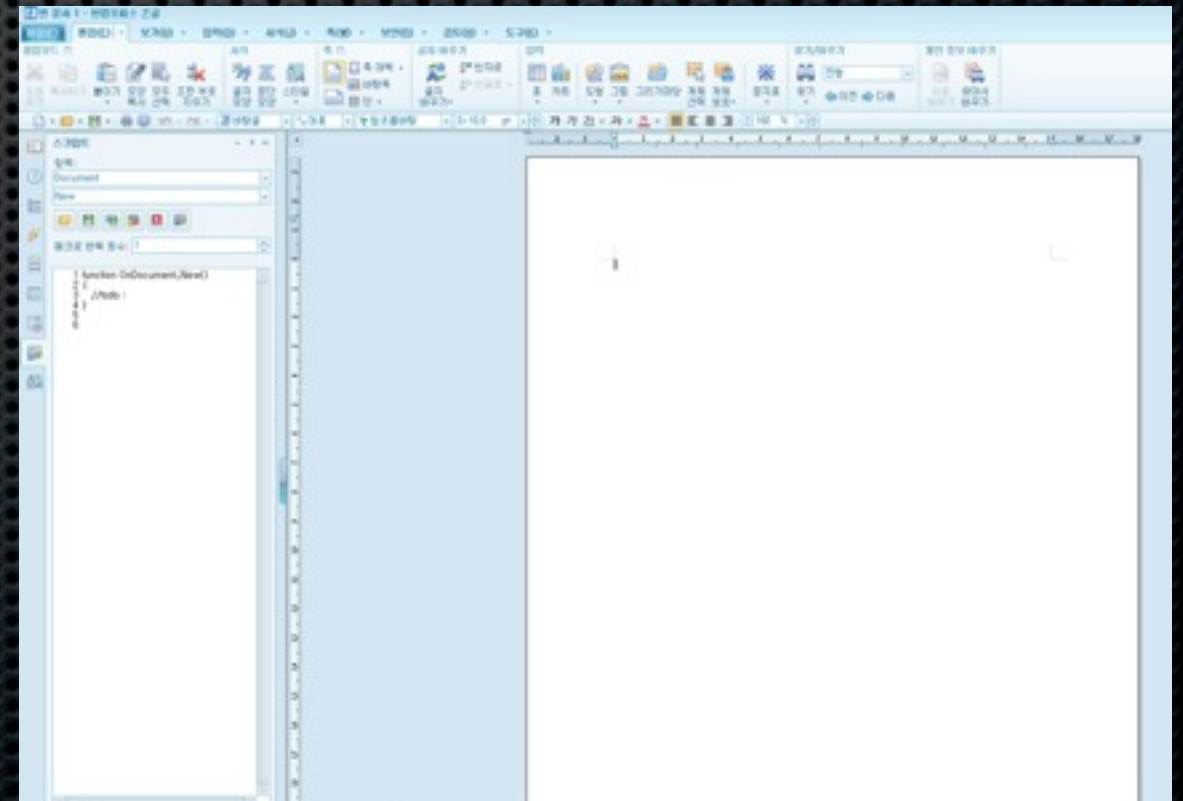
- ✦ No good code coverage as much as IL + Taint + Symbolic execution

Future work

- ✦ Again, this is very immature
- ✦ Reentrant issue: Change code like detours style
- ✦ Apply this way not only for file format but also network
- ✦ Combine symbolic execution at some stages
- ✦ Find more useful ways to apply different flows data
- ✦ Diff flows and change data of files automatically

Hangul HWP

- ✧ HWP, It is most popular word processor in Korea
- ✧ 99.9% government office *has* to use this program
 - It is mandatory, seriously
- ✧ Much more used than the MS word processor
- ✧ <http://www.hancom.com/>



APT on HWP

- ✦ HWP has been used for APT attacks in Korea
- ✦ Other formats have been used as well, but most of them were known 0days or 1days
- ✦ However, there are cases which used real HWP 0days for APT attacks
- ✦ Problem: any Anti-Virus detects malicious HWP **(only Hauri AV detects but it's based on signatures)**

Side story about HWP fuzzing

- ✦ Bug hunters know good samples are important to fuzz
- ✦ We made a script to collect .hwp files on the internet
- ✦ One thing came to my mind
- ✦ I have a friend who has a HWP expert certificate

Side story about HWP fuzzing

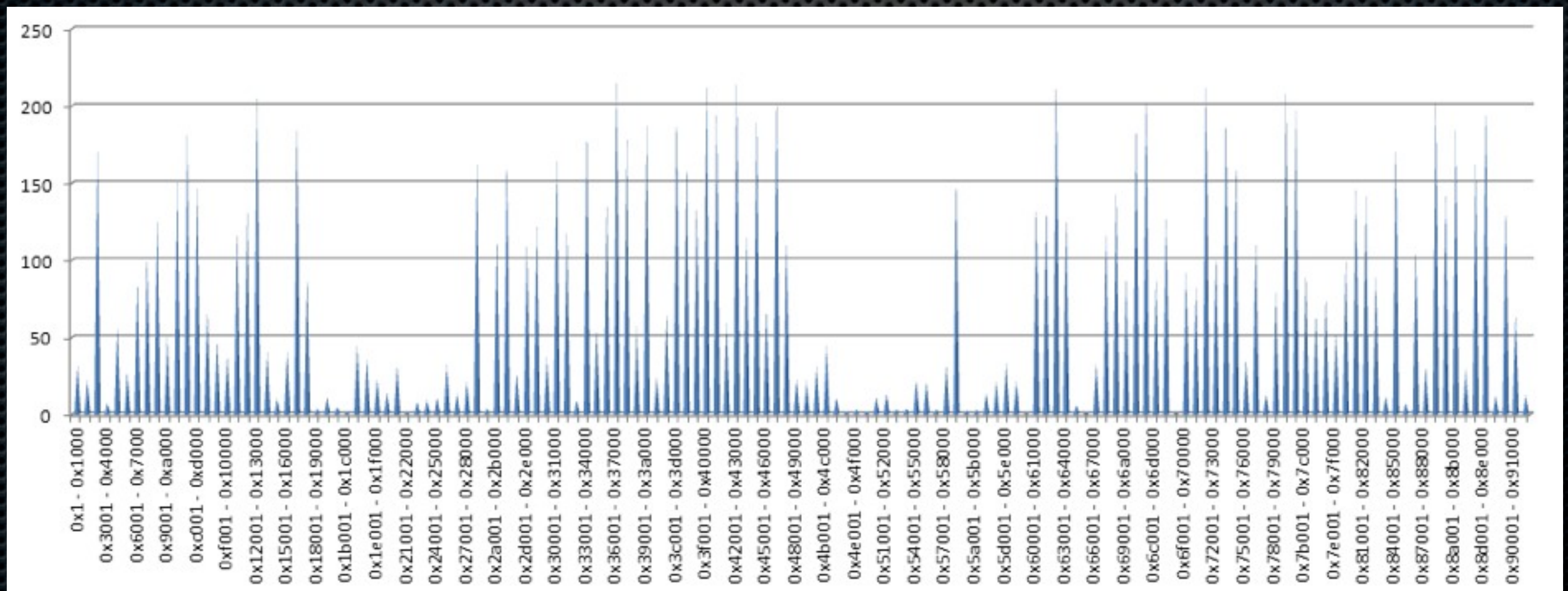


- ✦ We paid him \$5 per hour for making hwp files
- ✦ We got extremely nice samples with the cheap price!
- **he used most/not-well-known features of HWP**
- ✦ Good to not do fuzzing for crappy internet samples
- ✦ We think we saved a lot of time

The HWP bug

- ✦ We've been fuzzing a lot for HWP

[sample hwp for fuzzing]



which position affects flows?

The HWP bug

- ✦ Got many exploitable memory corruption bugs
 - We don't say the number!
- ✦ One of them will be used for our demo
- ✦ Multiply operation for a value (value * 4)
- ✦ $0x400000YY * 4$
- ✦ Sounds like exploitable obviously
- ✦ This has not been reported to the vendor yet

The HWP bug

```
mov ecx, [local value]  
mov edx, [ecx]  
mov eax, [edx+offset]  
call eax
```

- ✦ We can make that the local value is randomly referenced using the multiply operation
- ✦ heap-spray, obviously
- ✦ [24 24 24 24 shellcode] - “add al, 0x24” (like nop)
- ✦ Objects are mapped from 0x2000d000 in our machine
 - didn't check in other machines

The HWP bug

- ✧ HWP supports flash objects
- ✧ We wanted to embed a flash file for heap-spray
- ✧ But the bug gets triggered before loading flash objects
- ✧ We'll use big image files for our demo
- ✧ We think there would be some way to heap-spray without flash objects (by abusing the file format)

HWP Oday DEMO

‘company x’

- ✦ acquire a korean company
 - ✦ switch from ahnlab to norton
 - ✦ install fancy appliances

Putting it all together

- ✦ Using a hwp exploit on a live network
- ✦ will any of the tools/appliances detect it?
 - ✦ Norton
 - ✦ FireEye
 - ✦ Netwitness
 - ✦ bluecoat
 - ✦ damballa



Where does the data go?

老根(2273040292) 11:14:55
韩国网站数据库

人寿 医院 求职 公职考试 女性网 数据库~1000万

新到银行 3000w

接单各种数据单国内勿扰·有需要的请联系

Data of millions members from South Korean music website is for sale; tests are provided. Also for sale: over 10 billion pieces of second hand data. Contact QQ 450968290.

South Korean NEXON left-login universal gaming account, 14 RMB

South Korean NEXON right-login DF accounts for experiencing, no password protection (no bonding), 15 RMB

South Korean NEXON right-login DF accounts for experiencing, password protection with full information, no bonding, otp 45RMB

South Korean NEXON right-login DF accounts for experiencing, underage player accounts, 25 RMB

South Korean NEXON right-login official server accounts, 20 RMB

References

- Pydbg in Paimei: <http://www.openrce.org/downloads/details/208/PaiMei>
- Halvar Flake - Blackhat 2003
<http://www.blackhat.com/presentations/bh-federal-03/bh-fed-03-halvar.pdf>
- <http://code.google.com/p/ouspg/wiki/Radamsa>
- BitBlaze: <http://bitblaze.cs.berkeley.edu>
- <https://code.google.com/p/sulley/>
- <http://peachfuzz.sourceforge.net/>
- Fuzzing: Brute Force Vulnerability Discovery
- Gray Hat Python

Questions?

✖ ?